APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/723,967	11/26/2003	Joseph G. Laura	IDF 2584 (4000-I6100)	9521

28003   7590   10/02/2008
SPRINT
6391 SPRINT PARKWAY
KSOPHT0101-Z2100
OVERLAND PARK, KS 66251-2100

EXAMINER
WANG, BEN C

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
10/02/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

	Application No.	Applicant(s)	
Office Action Summary	10/723,967	LAURA, JOSEPH G.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS,
WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed
  after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any
  earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _6/16/08; 8/14/08._
2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.
3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is
closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-38_ is/are pending in the application.
   4a) Of the above claim(s) _____ is/are withdrawn from consideration.
5)☐ Claim(s) _____ is/are allowed.
6)☒ Claim(s) _1-38_ is/are rejected.
7)☐ Claim(s) _____ is/are objected to.
8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.
10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.
   Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
   Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a)☐ All b)☐ Some * c)☐ None of:
   1.☐ Certified copies of the priority documents have been received.
   2.☐ Certified copies of the priority documents have been received in Application No. _____.
   3.☐ Copies of the certified copies of the priority documents have been received in this National Stage
       application from the International Bureau (PCT Rule 17.2(a)).
   * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1)☒ Notice of References Cited (PTO-892)
2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3)☐ Information Disclosure Statement(s) (PTO/SB/08)
   Paper No(s)/Mail Date _____.
4)☐ Interview Summary (PTO-413)
   Paper No(s)/Mail Date. _____
5)☐ Notice of Informal Patent Application
6)☐ Other: _____.

DETAILED ACTION

1.   A request for continued examination under 37 CFR 1.114, including the fee set

forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this

application is eligible for continued examination under 37 CFR 1.114, and the fee set

forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action

has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on August

14, 2008 has been entered.

2.   Applicant's responsive to communication(s) filed on June 16, 2008 and August

14, 2008 to the Office action mailed April 16, 2008 provided in the rejection of claims 1-

38, wherein claims 1-4, 7-8, 10-12, 15-16, 18, 21, and 35-36 and have been amended.

        Claims 1-38 remain pending in the application and which have been fully

considered by the examiner.

        Applicant's arguments with respect to claims currently amended have been fully

considered but are moot in view of the new grounds of rejection – see *Nace et al.* - art made of

record, as applied hereto.

Claim Rejections – 35 USC § 102(e)

The following is quotation of 35 U.S.C. 102(e) which form the basis for all obviousness

rejections set forth in this office action:

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed
> in the United States before the invention by the applicant for patent or (2) a patent granted on an application for
> patent by another filed in the United States before the invention by the applicant for patent, except that an
> international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this

subsection of an application filed in the United States only if the international application designated the United
States and was published under Article 21(2) of such treaty in the English language.

3.   Claims 21-24, 26, 28, 30, and 35-37 are rejected under 35 U.S.C. 102(e) as

being anticipated by Nace et al. (Pub. No. US 2004/0268363 A1) (hereinafter 'Nace' -

art made of record)

4.   **As to claim 21** (Currently Amended), Nace discloses a system for non-

intrusively monitoring variables during operation of an application, comprising:

   o   a compile listing stored on a computer-readable medium having an address
       map with an offset associated with each of a plurality of variable of an
       application (e.g., Fig. 2, element – Mapping; element 122 – API to memory
       managers; element 118a – Memory Manager; [0008], Lines 13-15 – A
       communications engine may <u>map a logical address for the memory block to</u>
       <u>physical memory within its address space on a dynamic basis</u>; [0022] - …
       <u>variables, address offset or other memory mapping data</u> and other
       information related to the requesting process ... to read and write data, such
       as variable, arrays, tables …; [0039] - … by computation of the offset or other
       know relation to its associated buddy block or segment …); and

   o   a module stored on a computer-readable medium that performs reading of the
       compile listing and obtaining the offset of at least one of the plurality of
       variables of the application (e.g., Fig. 2, element – Mapping; element 122 –
       API to memory managers; element 118a – Memory Manager), the module
       performs attaching to an address space (e.g., [0008], Lines 13-15 – A

communications engine may <u>map a logical address for the memory block to physical memory within its address space on a dynamic basis</u>) used by the application during real-time operation to obtain a value for one or more the plurality of variables written to the address space by the application (e.g., Fig. 5, step 512; [0035] - ... that a memory block ... is available, processing may proceed to step 512 in which the initiating process, communications engine or other entity may populate the open block or sub-block with data to be communicated to a destination process ...) during the real-time operation of the application using the offset applications (e.g., [0022] - ... <u>variables, address offset or other memory mapping data</u> and other information related to the requesting process ... to read and write data, such as variable, arrays, tables ...)

5.   **As to claim 22** (Previously Presented), Nace discloses the system wherein the module is further operable to read the compile listing and convert at least one of the plurality of variables to the associated offset (e.g., [0022] - ... <u>variables, address offset or other memory mapping data</u> and other information related to the requesting process ... to read and write data, such as variable, arrays, tables ...)

6.   **As to claim 23** (Previously Presented), Nace discloses the system wherein the module is further operable to search the compile listing and display the plurality of variables of the application for selection by a user (e.g., Fig. 2, element – Mapping;

element 122 – API to memory managers; element 118a – Memory Manager; [0008],

Lines 13-15 – A communications engine may map a logical address for the memory

block to physical memory within its address space on a dynamic basis; [0022] - …

variables, address offset or other memory mapping data and other information related to

the requesting process ... to read and write data, such as variable, arrays, tables …;

[0039] - … by computation of the offset or other know relation to its associated buddy

block or segment …)

7.   **As to claim 24** (Original), Nace discloses the system wherein the module is

responsive to selection by the user of one of the plurality of variables to obtain the value

for the selected one of the plurality of variables using the offset to locate the value of the

variable in the address space (e.g., [0022] - … variables, address offset or other

memory mapping data and other information related to the requesting process ... to

read and write data, such as variable, arrays, tables …; [0039] - … by computation of

the offset or other know relation to its associated buddy block or segment …)

8.   **As to claim 26** (Original), Nace discloses the system wherein the address space

is further defined as a memory space and wherein the module attaches, using a socket

layer, to the memory space used by the application (e.g., Abstract, Lines 1-4 – An inter-

process communications platforms enables individual processes to request and

exchange data in a shared memory space ...)

9.   **As to claim 28** (Previously Presented), Nace discloses the system wherein the monitor is further operable, using the compile listing, to query the address map for one or more of the plurality of variables of the application (e.g., Fig. 2, element – Mapping; element 122 – API to memory managers; element 118a – Memory Manager; [0008], Lines 13-15 – A communications engine may <u>map a logical address for the memory block to physical memory within its address space on a dynamic basis</u>; [0022] - … <u>variables, address offset or other memory mapping data</u> and other information related to the requesting process … to read and write data, such as variable, arrays, tables …; [0039] - … by computation of the offset or other know relation to its associated buddy block or segment …)

10.   **As to claim 30** (Previously Presented), Nace discloses the system wherein the module is further operable to attach to the memory space where the application is operating and overwrite the value for one or more of the plurality of variables using the offset (e.g., [0022] - … <u>variables, address offset or other memory mapping data</u> and other information related to the requesting process … to read and <u>write data</u>, such as variable, arrays, tables …)

11.   **As to claim 35** (Currently Amended), Nace discloses a system for non-intrusively monitoring COBOL application values, the system comprising:

- a COBOL program (e.g., [0028] - … it will be appreciated that <u>different types of code or instruction may be used</u>) stored on a computer-readable medium that creates a

shared memory area through a technical layer (e.g., Fig. 2, element – Mapping;

element 122 – API to memory managers; element 118a – Memory Manager; [0008],

Lines 13-15 – A communications engine may <u>map a logical address for the memory</u>

<u>block to physical memory within its address space on a dynamic basis</u>; [0022] - …

<u>variables, address offset or other memory mapping data</u> and other information

related to the requesting process … to read and write data, such as variable, arrays,

tables …; [0039] - … by computation of the offset or other know relation to its

associated buddy block or segment …), generates program values and store the

program values in the memory area during real-time operation of the COBOL

program (e.g., Abstract, Lines 6-11 - … via an administrative memory space <u>which</u>

<u>tracks pointers, handles and other indicators of memory area populated by individual</u>

<u>processes</u>. When one process requests <u>access to a variable, pointer or other data</u>

<u>generated by another process</u>, the request is mediated by the communications

engine; [0022] - … <u>variables, address offset or other memory mapping data</u> and

other information related to the requesting process … to read and write data, such as

variable, arrays, tables …); and a COBOL monitor module stored on a computer-

readable medium that shares the memory area with the COBOL program through

the technical layer (e.g., Fig. 2, element 108 – Communication Engine; [0018] –

Communications engine 108 may in one regard mediate the exchange of data

between any process in the set of processes … with one or more other processes in

the same global set of processes, with other processes or network …; Abstract,

Lines 6-11 - … via an administrative memory space <u>which tracks pointers, handles</u>

and other indicators of memory area populated by individual processes. When one

process requests access to a variable, pointer or other data generated by another

process, the request is mediated by the communications engine) and the COBOL

monitor module reads the program values stored in the shared memory area by the

COBOL program during real-time operation of the COBOL program (e.g., Abstract,

Lines 6-11 - … via an administrative memory space which tracks pointers, handles

and other indicators of memory area populated by individual processes. When one

process requests access to a variable, pointer or other data generated by another

process, the request is mediated by the communications engine)

12.   **As to claim 36** (Currently Amended), Nace discloses the system further

comprising: a second COBOL program operable to generate second program values

and store the program values in the shared memory area during real-time operation of

the second COBOL program, and wherein the COBOL monitor module is further

operable to read the second program values stored in the shared memory area by the

second COBOL program (e.g., Fig. 2, element 108 – Communication Engine; [0018] –

Communications engine 108 may in one regard mediate the exchange of data between

any process in the set of processes … with one or more other processes in the same

global set of processes, with other processes or network …; Abstract, Lines 6-11 - …

via an administrative memory space which tracks pointers, handles and other indicators

of memory area populated by individual processes. When one process requests access

to a variable, pointer or other data generated by another process, the request is

mediated by the communications engine)

13.   **As to claim 37** (Original), Nace discloses the system further comprising: a

second memory area; and a second COBOL program operable to generate second

program values and store the program values in the second memory area during real-

time operation of the second COBOL program, and wherein the COBOL monitor

module is further operable to read the second program values stored in the second

memory area by the second COBOL program (e.g., Fig. 2, element 108 –

Communication Engine; [0018] – Communications engine 108 may in one regard

mediate the exchange of data between any process in the set of processes … with one

or more other processes in the same global set of processes, with other processes or

network …; Abstract, Lines 6-11 - … via an administrative memory space which tracks

pointers, handles and other indicators of memory area populated by individual

processes. When one process requests access to a variable, pointer or other data

generated by another process, the request is mediated by the communications engine)

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

14.   Claims 1-4, 6-20, and 38 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Nace in view of Sridharan et al. (*On Building Non-Intrusive

Performance Instrumentation Blocks for CORBA-based Distributed Systems, March

2000, IEEE*) (hereinafter 'Sridharan')

15.   **As to claim 1** (Currently Amended), Nace discloses a system for non-intrusively

monitoring an application, comprising:

- at least one application (e.g., Fig. 2, elements 104 – Shared Memory Space; 102

  – Process; [0015] - … a shared memory space 104 may be accessed by a global

  set of processes 102a, 102b … <u>containing an arbitrary number of software

  applications</u>) stored on a computer-readable medium, the at least one application

  creates a shared memory area (e.g., Fig. 4, step 412 – Create Memory Block;

  [0031] – In step 412, <u>a new or newly registered memory block</u> within the set of

  memory blocks … may be secured or <u>generated</u> (if not preexisting) …) and

  stores application values (e.g., Fig. 5, step 512; [0035] - … that a memory block

  … is available, processing may proceed to step 512 in which the initiating

process, communications engine or other entity may populate the open block or sub-block with data to be communicated to a destination process ...) in the shared memory area (e.g., Fig. 2, elements 104 – Shared Memory Space);

- a first module (e.g., Abstract, Lines 1-3 – An inter-process communications platform enables individual process <u>to request and exchange data in a shared memory space</u>, mediated by a communication engine) stored on a computer-readable medium that shares and attaches to the shared memory area that is used by the at least one application during real-time operation, the first module reads application values from the shared memory area that have been stored in the shared memory area by at least one the application during real-time operation (e.g., Abstract, Lines 6-11 - ... via an administrative memory space <u>which tracks pointers, handles and other indicators of memory area populated by individual processes</u>. When one process requests <u>access to a variable, pointer or other data generated by another process</u>, the request is mediated by the communications engine);

- a second module stored on a computer-readable medium in communication with the first module that requests the first module to read the application values, the second module receives the application values from the first module (e.g., Fig. 2, element 108 – Communication Engine; [0018] – Communications engine 108 may in one regard mediate the exchange of data between any process in the set of processes ... with one or more other processes in the same global set of processes, with other processes or network ...; Abstract, Lines 6-11 - ... via an

administrative memory space <u>which tracks pointers, handles and other indicators</u> <u>of memory area populated by individual processes.</u> When one process requests <u>access to a variable, pointer or other data generated by another process</u>, the request is mediated by the communications engine);

Further, Nace discloses software platforms designed to enable applications or other processes <u>executing in a shared memory space</u> <u>to exchange data on an secure and</u> <u>efficient basi</u>s (e.g., [0003]) but does not explicitly disclose a third module stored on a computer-readable medium in communication with the second module, that displays the application values.

However, in an analogous art of *On Building Non-Intrusive Performance Instrumentation Blocks for CORBA-based Distributed Systems*, Sridharan discloses a third module stored on a computer-readable medium in communication with the second module, that displays the application values (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PM GUI"; P. 3, last Par. – A GUI was implemented to display the various performance data, to set the filtering information and turn on and off the monitoring mechanism)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Sridharan into the Nace's system to further provide a third module stored on a computer-readable medium in communication with the second module, that displays the application values s in the Nace system.

The motivation is that it would further enhance the Nace's system by taking,

advancing and/or incorporating the Sridharan's system which offers significant

advantages for the utility of the framework in quantifying the benefits of a performance

improvement technique and identifying performance-bottlenecks in the system as once

suggested by Sridharan (e.g., Sec. 7 – Concluding Remarks)

16.   **As to claim 2** (Currently Amended), Nace discloses the system wherein the

shared memory area is further defined as a shared memory of the application (e.g., Fig.

2, elements 104 – Shared Memory Space; 102 – Process; [0015] - ... a shared memory

space 104 may be accessed by a global set of processes 102a, 102b ... containing an

arbitrary number of software applications)

17.   **As to claim 3** (Currently Amended), Nace discloses the system wherein the first

module is further operable to attach to the shared memory area used by the at least one

application to read the application values (e.g., Abstract, Lines 6-11 - ... via an

administrative memory space which tracks pointers, handles and other indicators of

memory area populated by individual processes. When one process requests access to

a variable, pointer or other data generated by another process, the request is mediated

by the communications engine)

18.   **As to claim 4** (Currently Amended), Nace discloses the system wherein the

application values are further defined as at least one application variable and a value for

at least one the application variable (e.g., Abstract, Lines 6-11 - ... via an administrative
memory space <u>which tracks pointers, handles and other indicators of memory area
populated by individual processes</u>. When one process requests <u>access to a variable,
pointer or other data generated by another process</u>, the request is mediated by the
communications engine)

19.   **As to claim 6** (Original), Sridharan discloses the system wherein the third
module is further defined as a graphical user interface (e.g., Sec. IV. – Designing A Tool
Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for
mapping each memory location observed by the monitoring system to its corresponding
program data structure identifier (procedure and variable names) is being implemented
as well).

20.   **As to claim 7** (Currently Amended), Sridharan discloses the system wherein the
graphical user interface is further operable to receive an input identifying the application
values to be read and operable to request the application values identified to the first
module (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PI"; Fig.
4 – Performance Instrumentation (PI) Module – White Box View, element "I"; P. 3, 7th
Par – In Fig. 4, the thread labeled T1 posts the reception of the request P(P_1,P_2) as an
event in the event service; P. 3, 1st Par. – The Visibroker ORB, which was used to
deploy the servers, provided the facility to load the Performance Instrumentation (PI)
module to together with the server into a single address space while staring the server

using the Java VM; P. 3, 2nd Par. – 4th Par.), via the second module (e.g., Fig. 3 –

System T and Performance Instrumentation, element of "ORB"; Sec. 5 –

Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows

interception of a CORBA call made to a server either at the client or server side; the

interceptor facility is a CORBA implementation or ORB's feature), and wherein the first

module is operable to read the requested application values data from the shared

memory area and return the application variables to the graphical user interface (e.g.,

Fig. 3 – System T and Performance Instrumentation, element of "PM GUI"; P. 3, last

Par. – A GUI was implemented to display the various performance data, to set the

filtering information and turn on and off the monitoring mechanism), via the second

module.

21.   **As to claim 8** (Original), Sridharan discloses the system wherein the graphical

user interface is further operable to receive an input identifying requested application

values to be displayed (e.g., Sec. 2, 4th Para. – cooperates with Graphical User

Interface in the process of information visualization, implements the selective monitoring

policy, and manages domains of local monitors; Sec. 5, 1st Para., 2nd Para. – GUI's

main role is visualization of behavior of monitored systems' selected parts)

22.   **As to claim 9** (Original), Sridharan discloses the system wherein the first module

is further operable as a socket server and wherein the second module is further

operable as a socket client such that the first and second modules communicate via a

socket connection (e.g., Fig. 3 – System T and Performance Instrumentation, element

of "PM GUI"; P. 3, last Par. – A GUI was implemented to display the various

performance data, to set the filtering information and turn on and off the monitoring

mechanism)

23.   **As to claim 10** (Currently Amended), Nace discloses the system wherein the

first module reads application values stored in the shared memory area by the at least

one application while the at least one application is running (e.g., Abstract, Lines 6-11 -

… via an administrative memory space <u>which tracks pointers, handles and other</u>

<u>indicators of memory area populated by individual processes</u>. When one process

requests <u>access to a variable, pointer or other data generated by another process</u>, the

request is mediated by the communications engine)

24.   **As to claim 11** (Currently Amended), Nace discloses the system wherein first

module operable to read application values stored in the shared memory area by the at

least one application without interfering with the operation of the at least one application

(e.g., Abstract, Lines 6-11 - … via an administrative memory space <u>which tracks</u>

<u>pointers, handles and other indicators of memory area populated by individual</u>

<u>processes</u>. When one process requests <u>access to a variable, pointer or other data</u>

<u>generated by another process</u>, the request is mediated by the communications engine)

25.   **As to claim 12** (Currently Amended), Nace discloses a method of non-intrusively

monitoring operation of an application, comprising:

- running an application in a real-time manner (e.g., Fig. 2, elements 104 – Shared
  Memory Space; 102 – Process; [0015] - … a shared memory space 104 may be
  accessed by a global set of processes 102a, 102b … <u>containing an arbitrary
  number of software applications</u>);

- creating a memory area (e.g., Fig. 4, step 412; [0031] – In step 412, <u>a new or
  newly registered memory block</u> within the set of memory blocks … may be
  secured or <u>generated</u> (if not preexisting) …);

- generating, by the application, application values during operation of the
  application (e.g., Fig. 5, step 512; [0035] - … that a memory block … is available,
  processing may proceed to step 512 in which the initiating process,
  communications engine or other entity may populate the open block or sub-block
  with data to be communicated to a destination process ...);

- writing, by the application, the application values in the memory area during the
  operation of the application (e.g., Fig. 5, step 512; [0035] - … that a memory
  block … is available, processing may proceed to step 512 in which the initiating
  process, communications engine or other entity may populate the open block or
  sub-block with data to be communicated to a destination process ...);

- reading, by a monitor, the memory area used by the application to obtain the
  application values, wherein at least one of the application values is not output by
  the application (e.g., Abstract, Lines 6-11 - … via an administrative memory

space <u>which tracks pointers, handles and other indicators of memory area</u>

<u>populated by individual processes</u>. When one process requests <u>access to a</u>

<u>variable, pointer or other data generated by another process</u>, the request is

mediated by the communications engine);

Further, Nace discloses software platforms designed to enable applications or other

processes <u>executing in a shared memory space</u> <u>to exchange data on an secure and</u>

<u>efficient basi</u>s (e.g., [0003]) but does not explicitly disclose displaying the application

values read from the memory area.

However, in an analogous art of *On Building Non-Intrusive Performance*

Instrumentation Blocks for CORBA-based Distributed Systems, Sridharan displaying the

application values read from the memory area (e.g., Fig. 3 – System T and Performance

Instrumentation, element of "PM GUI"; P. 3, last Par. – A GUI was implemented to

display the various performance data, to set the filtering information and turn on and off

the monitoring mechanism)

Therefore, it would have been obvious to one of ordinary skill in the art, at the

time the invention was made to combine the teachings of Sridharan into the Nace's

system to further provide displaying the application values read from the memory area

in the Nace system.

The motivation is that it would further enhance the Nace's system by taking,

advancing and/or incorporating the Sridharan's system which offers significant

advantages for the utility of the framework in quantifying the benefits of a performance

improvement technique and identifying performance-bottlenecks in the system as once

suggested by Sridharan (e.g., Sec. 7 – Concluding Remarks)

26.   **As to claim 13** (Previously Presented), Sridharan discloses the method further

comprising: requesting, by a client, application values from the monitor (e.g., Fig. 3 –

System T and Performance Instrumentation, element of "PI"; P. 3, 1st Par. – The

Visibroker ORB, which was used to deploy the servers, provided the facility to load the

Performance Instrumentation (PI) module to together with the server into a single

address space while staring the server using the Java VM; P. 3, 2nd Par. – 4th Par.); and

communicating the application variables from the monitor to the client (e.g., Fig. 3 –

System T and Performance Instrumentation, element of "ORB"; Sec. 5 –

Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows

interception of a CORBA call made to a server either at the client or server side; the

interceptor facility is a CORBA implementation or ORB's feature)

27.   **As to claim 14** (Previously Presented), Nace discloses the method further

comprising: requesting application values; running a plurality of applications in a real-

time manner; generating application values stored in one or more memory areas during

operation of the plurality of applications; reading the one or more memory areas used

by the plurality of applications to obtain the application values (e.g., Fig. 2, elements

104 – Shared Memory Space; 102 – Process; [0015] - … a shared memory space 104

may be accessed by a global set of processes 102a, 102b … <u>containing an arbitrary</u>

number of software applications; Fig. 5, step 512; [0035] - ... that a memory block ... is

available, processing may proceed to step 512 in which the initiating process,

communications engine or other entity may populate the open block or sub-block with

data to be communicated to a destination process ...; Abstract, Lines 1-3 – An inter-

process communications platform enables individual process to request and exchange

data in a shared memory space, mediated by a communication engine); and Sridharan

discloses displaying the requested application values (e.g., Fig. 3 – System T and

Performance Instrumentation, element of "ORB"; Sec. 5 – Instrumentation Framework,

1st Par. – The CORBA Interceptor facility allows interception of a CORBA call made to a

server either at the client or server side; the interceptor facility is a CORBA

implementation or ORB's feature)

28.   **As to claim 15** (Currently Amended), Nace discloses the method wherein the

memory area is further defined as a block of shared memory (e.g., Fig. 4, step 412 –

Create Memory Block; [0031] – In step 412, a new or newly registered memory block

within the set of memory blocks ... may be secured or generated (if not preexisting) ...)

and wherein the monitor reads at least some of the application variables stored in the

block of memory (e.g., Abstract, Lines 6-11 - ... via an administrative memory space

which tracks pointers, handles and other indicators of memory area populated by

individual processes. When one process requests access to a variable, pointer or other

data generated by another process, the request is mediated by the communications

engine)

29.   **As to claim 16** (Currently Amended), Nace discloses the method further

comprising providing a memory manager and wherein the monitor registers with the

memory manager to obtain a location of the memory area used by the application to

store the application values (e.g., Fig. 2, elements – 118a, 118b; [0025] - ... With

access to address translation, memory mapping or other data for each process in the

set of processes ... the set of memory managers ... in administrative memory may

ensure coherency in the set of memory block ...)

30.   **As to claim 17** (Original), Nace discloses the method further comprising:

generating new application values by the application stored in the memory area, at least

one of the new application values defined as a new value for a variable of the

application; requesting, by the client, that the monitor re-read the application values

stored in the memory area; re-reading, by the monitor, the memory area to obtain the

new application values (e.g., Abstract, Lines 6-11 - ... via an administrative memory

space which tracks pointers, handles and other indicators of memory area populated by

individual processes. When one process requests access to a variable, pointer or other

data generated by another process, the request is mediated by the communications

engine)

31.   **As to claim 18** (Currently Amended), Nace discloses the method wherein the

monitor reads the application values while the application is running (e.g., Abstract,

Lines 6-11 - ... via an administrative memory space <u>which tracks pointers, handles and</u>
<u>other indicators of memory area populated by individual processes</u>. When one process
requests <u>access to a variable, pointer or other data generated by another process</u>, the
request is mediated by the communications engine)

32.   **As to claim 19** (Original), Sridharan discloses the method wherein the monitor is
operable as a socket server and wherein the client is operable as a socket client such
that the communication between the monitor and client is via a socket connection (e.g.,
Fig. 3 – System T and Performance Instrumentation, element of "ORB"; Sec. 5 –
Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows
interception of a CORBA call made to a server either at the client or server side; the
interceptor facility is a CORBA implementation or ORB's feature)

33.   **As to claim 20** (Original), Nace discloses the method wherein the application
values are further defined as a variable of the application and a value of the variable
(e.g., Abstract, Lines 6-11 - ... via an administrative memory space <u>which tracks</u>
<u>pointers, handles and other indicators of memory area populated by individual</u>
<u>processes</u>. When one process requests <u>access to a variable, pointer or other data</u>
<u>generated by another process</u>, the request is mediated by the communications engine)

34.   **As to claim 38** (Original), Sridharan discloses the system further comprising: a
user interface operable to monitor and display the application values; and a client

application in communication with the user interface and the COBOL monitor module,

the client application operable to request the program variables of the COBOL program

from the COBOL monitor module and provide the program variables to the user

interface for display via the user interface responsive to a request from the user

interface (e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE

Monitoring Approach", 5th Par. – A mechanism for mapping each memory location

observed by the monitoring system to its corresponding program data structure identifier

(procedure and variable names) is being implemented as well)

35.   Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nace in

view of Sridharan and further in view of Hiroshi Kashima (*An Approach for Constructing

Web Enterprise Systems on Distributed Objects, Jan., 2000, IBM*) (hereinafter

'Kashima')

36.   **As to claim 5** (Original), Nace and Sridharan do not disclose the system wherein

the first module is further operable to communicate the application values to the second

module in hypertext markup language format.

        However, in an analogous art of *an approach for constructing web enterprise

systems on distributed objects*, Kashima discloses the system wherein the first module

is further operable to communicate the application values to the second module in

hypertext markup language format (i.e., Abstract, 1st Para.; Sec. 1-2, 1st Para.; Sec. 1-

3, 5th Para., Lines 1-3)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Nace-Sridharan and the teachings of Kashima to further provide the system wherein the first module is further operable to communicate the application values to the second module in hypertext markup language format in Nace-Sridharan system.

The motivation is that it would enhance the Nace-Sridharan system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support')

37.   Claims 25 and 31-32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nace in view of Jie Tao et al. *(Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures, Springer-Verlag Berlin Heidelberg 2001, pp. 861-870)* (hereinafter 'Tao-2')

38.   **As to claim 25** (Original), Nace discloses software platforms designed to enable applications or other processes executing in a shared memory space to exchange data on an secure and efficient basis (e.g., [0003]) but does not explicitly disclose the system wherein the module is further operable to display the selected one of the plurality of variables.

However, in an analogous art of *Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures,* Tao-2 discloses the system wherein the module is further operable to display the selected one of the plurality of variables (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - ... additional tools are necessary in order <u>to transform the user-unreadable monitored data in a more understandable and easy-to-use form</u> ...; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - ... the visualizer offers a "Data structure" window <u>to reflect this mapping</u> .... <u>Shows all the shared variables occurring in a source code</u> ... provides users with a global overview of accesses to the complete working set of the application ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Tao-2 into the Nace's system to further provide the system wherein the module is further operable to display the selected one of the plurality of variables in the Nace system.

The motivation is that it would further enhance the Nace's system by taking, advancing and/or incorporating Tao-2's system which offers significant advantages that presents such a visualization tool displaying the monitored data in a user understandable way thereby showing the memory access behavior of shared memory applications as once suggested by Tao-2 (e.g., Abstract, Lines 8-15)

39.   **As to claim 31** (Previously Presented), Tao-2 discloses the system wherein the module comprises: a reader component operable to perform reading the compile listing

and further operable to perform converting at least one of the plurality of variables of the

application to the associated offset; and a search component that performs receiving

the associated offset of the at least one of the plurality of variables from the reader

component, the search component operable to perform attaching to the application and

further operable to locate the value of the at least one of the plurality of variables (e.g.,

Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory

programming; Sec. 1 – Introduction, 5th Par. - ... additional tools are necessary in order

to transform the user-unreadable monitored data in a more understandable and easy-to-

use form ...; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - ... the visualizer

offers a "Data structure" window to reflect this mapping .... Shows all the shared

variables occurring in a source code ... provides users with a global overview of

accesses to the complete working set of the application ...); and

Nace discloses using the offset (e.g., [0022] - ... variables, address offset or

other memory mapping data and other information related to the requesting process ...

to read and write data, such as variable, arrays, tables ...)

40.   **As to claim 32** (Previously Presented), Tao-2 discloses the system further

comprising a display component operably coupled to the module to perform receiving

the value for the one or more of the plurality of variables, the display component

operable to perform displaying the value (e.g., Fig. 1 – Infrastructure of the on-line

monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - ...

additional tools are necessary in order to transform the user-unreadable monitored data

in a more understandable and easy-to-use form …; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - … the visualizer offers a "Data structure" window to reflect this mapping …. Shows all the shared variables occurring in a source code … provides users with a global overview of accesses to the complete working set of the application …)

41.   Claims 27 and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nace in view of Huang et al., (*Operating System Support for Flexible Coherence in Distributed Shared Memory, 1996, IEEE*) (hereinafter 'Huang')

42.   **As to claim 27** (Original), Nace does not disclose the system wherein the module attaches, using the offset, to the memory space used by the application via an operating system service.

However, in an analogous art of *Operating System Support for Flexible Coherence in Distributed Shared Memory*, Huang the system wherein the module attaches, using the offset, to the memory space used by the application via an operating system service (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object; Sec. 4.3 – Programming Interface, 1st Par. – *AcquireLock*()– acquires a lock for an object fragment specified by object, offset and length, *ReleaseLock*())

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Huang into the Nace's system to further provide the system wherein the module attaches, using the offset, to the memory space used by the application via an operating system service in Nace's system.

The motivation is that it would further enhance the Nace's system by taking, advancing and/or incorporating Huang's system which offers significant advantages that the major advantages are the openness and ability to enforce modularity behind memory protection boundaries, sometimes characterized as the separation of policy from mechanism as once suggested by Huang (e.g., Sec. 3.1 - Microkernels)

43.   **As to claim 29** (Original), Huang discloses the system wherein the module is further defined as a subtask of the operating system (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object; Sec. 4.3 – Programming Interface, 1st Par. – *AcquireLock()* – acquires a lock for an object fragment specified by object, offset and length, *ReleaseLock()*)

44.   Claims 33-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nace in view of Tao-2 and further in view of Tao et al. (*Understanding the Behavior of Shared Memory Applications Using the SMiLE Monitoring Framework, March 2000, IEEE*) (hereinafter 'Tao-1')

45.   **As to claim 33** (Original), Nace and Tao-1 do not explicitly disclose the system
wherein the display component is operable to employ the value to display a heartbeat.

However, in an analogous art of *Understanding the Behavior of Shared Memory
Applications Using the SMiLE Monitoring Framework*, Tao-1 discloses the system
wherein the display component is operable to employ the value to display a heartbeat
(e.g., Sec. I – Motivation, 3rd Par., Lines 18-26 – it provides the programmer or system
software with detailed information about all memory transactions over the network and
the behavior of user-defined events)

Therefore, it would have been obvious to one of ordinary skill in the art, at the
time the invention was made to combine the teachings of Tao-1 into the NaceTao_2's
system to further provide the system wherein the display component is operable to
employ the value to display a heartbeat in the Nace-Tao_2' system.

The motivation is that it would further enhance the NaceTao_2's system by
taking, advancing and/or incorporating Tao-1's system which offers significant
advantages that it provides the programmer or system software with detailed
information about all memory transactions over the network and the behavior of user-
defined events as once suggested by Tao-1 (e.g., Sec. I. – Motivation, 3rd Par., Lines
18-26)

46.   **As to claim 34** (Original), Tao-1 discloses the system wherein the display
component is operable to employ the value to display as a percentage complete

heartbeat (e.g., Sec. I – Motivation, 3rd Par., Lines 18-26 – it provides the programmer

or system software with detailed information about all memory transactions over the

network and the behavior of user-defined events)

Conclusion

47.   Any inquiry concerning this communication or earlier communications from the

examiner should be directed to Ben C. Wang whose telephone number is 571-270-

1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00

p.m., EST.

    If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for

the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the

Patent Application Information Retrieval (PAIR) system. Status information for

published applications may be obtained from either Private PAIR or Public PAIR.

Status information for unpublished applications is available through Private PAIR only.

For more information about the PAIR system, see http://pair-direct.uspto.gov. Should

you have questions on access to the Private PAIR system, contact the Electronic

Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a

USPTO Customer Service Representative or access to the automated information

system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/   /Tuan Q. Dam/
Examiner, Art Unit 2192   Supervisory Patent Examiner, Art Unit 2192